

## Automatic Form Filling Based on Ontology-Controlled Dialogue With the User

Łukasz Bownik, Wojciech Górka, Adam Piasecki

Centrum Elektryfikacji i Automatykacji Górnictwa "EMAG"  
ul. Leopolda 31, 40-189 Katowice  
{lownik, wgorka, apiasecki}@emag.pl

**Abstract.** Semantic Web is now one of the most common directions in the IT-oriented research. Here, the focus is put on issues related to the logical layer of semantic applications, i.e. inference methods within constantly broadening ranges of logic as well as standardization of successive languages with increasing logical expressiveness. The article features the solution to the issue of automatic form filling with the use of ontology-controlled dialogue with the user. This solution resulted in the implementation of a universal programming module. Practical application of semantic technologies based on previous achievements in the field, combining the possibilities of inference machines with the flexibility of the RDF language, allows to achieve certain benefits for the users of an IT system. The Authors assume that the reader possesses some basic knowledge in the range of ontology structure as well as the RDF and OWL languages.

### 1 Introduction

The basic issue connected with the initiation of public services is the necessity to fill many forms in which certain data are repeated. Most forms require to provide the same data (similar data sets or data dependent on other data included in other parts of the form) whose filling is connected with mental discomfort for the user. The user has to enter the same things again and again. The work described in this paper aimed at developing software that would support the user in the process of filling e-forms through automatic and pseudo-intelligent filling the fields of a form with the information possessed by an IT system. Basic assumptions were the following:

- Modular structure of the software which will enable integration with any IT system,
- Full independence from the domain the form refers to so the software wouldn't include any assumptions related to the meaning and format of the processed data;
- Minimum number of interactions with the user which is both the objective of the work and the qualitative measure of the developed solution.

The result of the conducted research is software in the form of a software library which supports the filling of e-forms based on the dialogue with the user. In the further part of the paper the Authors will show the results of their work, the description of the architecture and the functioning of the developed software, henceforth called a "dialogue module".

Due to a highly interactive nature of the dialogue module and the limited space of the paper it was not possible to illustrate the functioning of the dialogue module by means of printed screens from the application which uses the dialogue module. Thus in order to demonstrate how the software works the Authors described an example of a user-system interaction while filling a form.

## 2 Related work

Similar solutions were proposed, among others, in the “RoboForm” [9] and “Mojo Navigator” [10] project but they focus, first of all, on the functionality of passwords wallet integrated with a web browser and having at its disposal the history of values associated with the form fields available on websites. There are also many solutions incorporating a dialog-based method of form filling. However, most of them are strictly based on a hard coded dialogue related to a single concrete form [11].

## 3 Ontologies

In order to understand the operations of the dialogue module it is necessary to grasp the data ontology and the form ontology ideas which lie at the basis of the process.

### 3.1 Data Ontologies

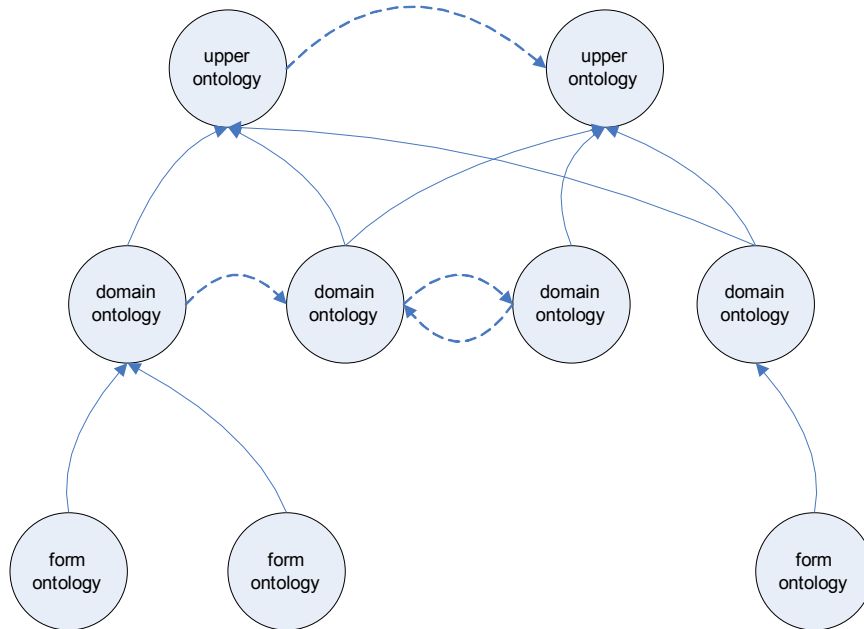
Data ontologies define the vocabulary (concepts) used in the course of data exchange between systems. Data ontology is not meant to organize knowledge (which is the case of expert systems) but to determine concepts describing the data which can be understood by many systems. In terms of their concept, data ontologies correspond with XML diagrams and are their functional equivalents.

The first and highest level (Fig.1) contains definitions of the most general vocabulary (concepts) commonly used by many IT systems (e.g. *person, age, name, family name* ). The concepts defined at this level are the basis for the definitions of more specialized concepts placed at lower levels of abstraction. **General ontologies should not (must not, to be precise) impose any restrictions** (e.g. in the form of limits as to the format of field value or number of fields in the form) on the defined concepts because each restriction placed at this level of generalization will diminish the universal force of the defined vocabulary and will raise the probability of conflicts with restrictions defined at lower levels (redundant or contrary restrictions). General ontologies must have total horizontal coherence, i.e. mutual coherence within the entire level of abstraction.

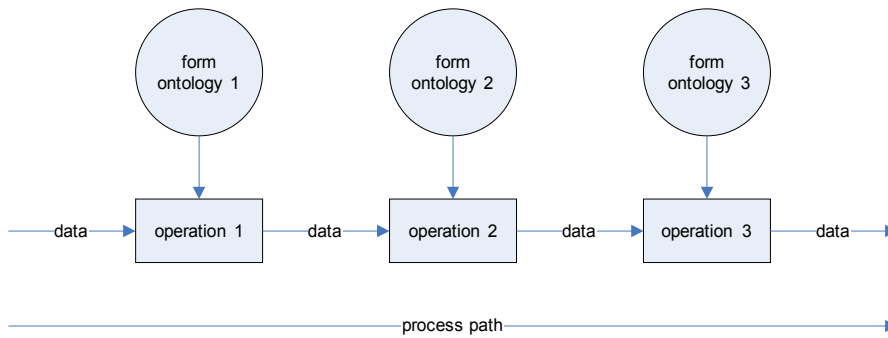
The second, middle level contains definitions of concepts shared by IT systems which operate within a given domain (e.g. medicine). These ontologies should use, to the highest possible extent, the concepts defined at a higher level of abstraction (i.e. they should not define their own concepts indicating, for example, *family name* ) as well as add the concepts from this very domain (e.g. *clinic, doctor, vaccine* ). At this

level of abstraction **only most obvious restrictions** should be defined, i.e. the restrictions which will be forced by all existing and future IT systems operating within a given domain. If it is not possible to explicitly define such restrictions, one should not define them at all because, as it is the case with general ontologies, they may provoke conflicts between ontologies. Domain ontologies must have horizontal coherence within a certain domain, i.e. mutual coherence within each domain at this level of abstraction as well as coherence with general ontologies.

The third and lowest level contains definitions of forms, i.e. specific data packages exchanged between IT systems. A form is a structure of data exchanged between two IT systems. The structure may, but does not have to, be related to the form filled by the system user. Each form there should have a form ontology describing its structure. The ontologies of the form should use, to the highest possible extent, the concepts defined at higher levels of abstraction and **define only such new concepts which are specific to a given form** and cannot be shared with other forms. The form ontology should define all necessary restrictions which will ensure logical coherence and correctness of the form required by the specifications of the IT system. Since for each form there are different rules (restrictions) which determine its coherence and correctness (e.g. one form requires a ID number, the other does not), it is not advisable to define restrictions at higher levels of abstraction. The form ontologies do not have to demonstrate any horizontal coherence (it is practically impossible to provide it). A logical error resulting from horizontal incoherence will never occur because there is separate processing of each ontology in each processing path in the IT system. The form ontologies must be coherent with the applied domain ontologies and general ontologies.



**Fig. 1.** Layered-tree ontology structure.



**Fig. 2 .** The overview of physical separation of processing form ontologies within single processing path.

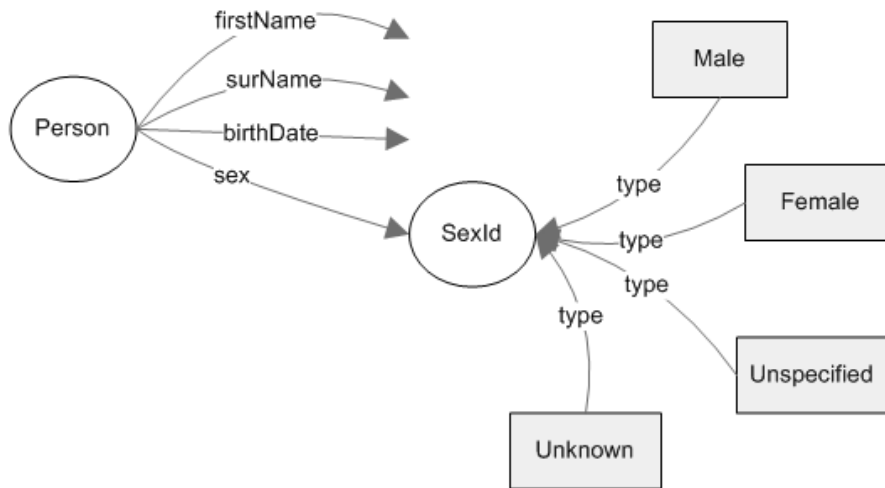


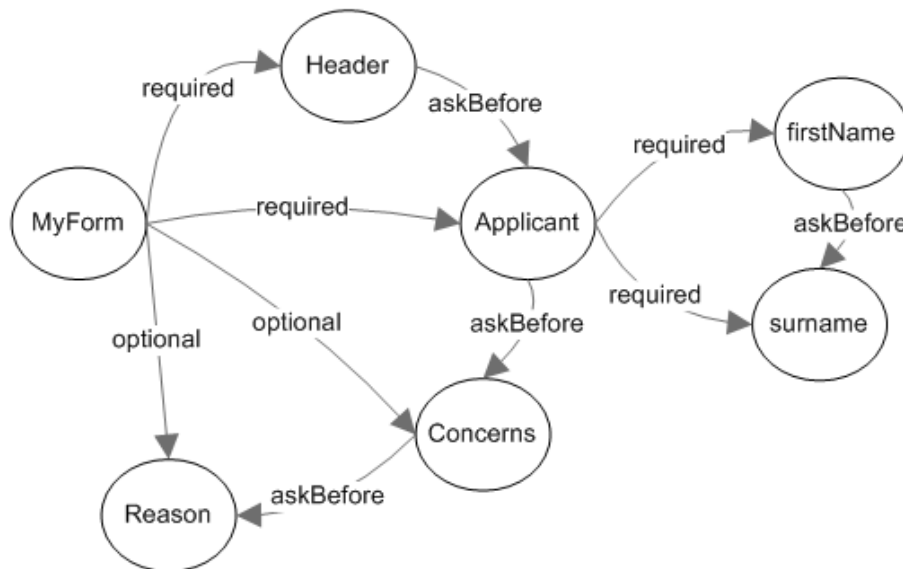
Fig. 3. Example of data ontology.

### 3.2 Form Ontologies

Form ontologies define the structure as well as coherence and correctness rules of data contained in the described form. A form ontology describes a specific set of data exchanged between IT systems.

The process of constructing a form ontology begins from building the main form class which will lie at the roots of the whole structure. The main class of the form must be a subclass of the Form class. Thanks to this assumption, it is possible to automatically detect the root of the form structure by searching the most specialized subclass of the Form class. The definition of the successive sections of the form is carried out by defining separate classes (e.g. Person class or Address class) linked with object properties to form a tree structure. The definitions of data fields are carried out with the use of datatype properties of classes which make up the sections of the form.

As the form ontology is based on concepts defined in data ontologies which may be much more extensive than the form requires, the *required* and *optional* properties were defined. They connect the desired properties with the classes being the sections of the form (see Fig. 4). In order to force the order of processing the sections and fields of the form, it is necessary to define the processing order for each pair of properties, representing these sections, by means of the *askBefore* property (see Fig. 4).



**Fig. 4.** A piece of example form ontology.

In order to define the desired restrictions it is necessary to determine new concepts due to the fact that the built-in cardinality concepts have their own defined semantics which does not suit the issue to be solved (first of all, the Open World Assumption, explained further in the article, and no possibility to mark the property as optional).

Form ontologies can contain rules (implications) that extend the functionality of the OWL language [4]. In order to secure logical non-contradiction, the rules must belong to the “DL-safe” set [6, 8]. The most frequent application of the rules is the structure of section labels (enabling, for example, creation of a section label for personal data on the basis of the entered name and surname) and automatic assigning of values to certain fields based on the values of other fields.

## 4 Character istics of the Dialogue Module

### 4.1 The Dialogue Module Functionality

The dialogue module works as an information filter. Having a set of input data (which can be empty) and a form ontology, the module “tries” to fill the form with the input data and the data provided by the user. Filling the form is done as a dialogue with the user through minimum interactions with the user assuming that the user is asked questions only if the data are unavailable or doubtful. The usability of the module, measured based on the number of interactions with the user (the fewer the better),

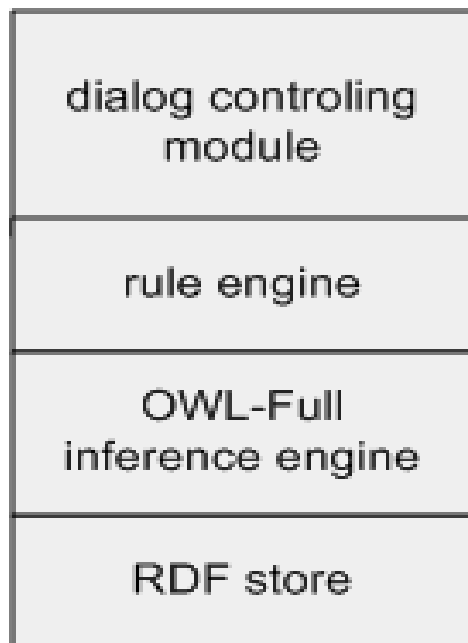
increases proportionally to the volume of input data available in the system. An example of the dialogue functioning will be described in the further part of the module.

#### 4.2 Dialogue module architecture

The dialogue module consists of four basic parts:

- the RDF base enabling access to data and ontologies (now Jena <http://jena.sourceforge.net/>),
- the OWL-Full inference machine responsible for the classification of concepts (now Pellet <http://pellet.owldl.com/>);
- a classic forward chaining inference engine which processes the definitions of extra rules of the form (now Jena);
- a module that controls the form filling process developed as a result of the conducted work (implemented in Java).

The figure below presents a layer structure of the dialogue module. For the module to function properly it is also necessary to have input data stored in the user's profile which was not shown in the figure.



**Fig. 5.** Dialogue module structure

The efficiency of the dialogue module depends on input data stored in the user's profile. In order to maximize the modular nature it was assumed that the input data are provided as an RDF graph in the process of initializing the dialogue module irrespective of the way they are stored in the system.

### 4.3 The Dialogue Module Operating Principle

The operating principle of the module is based on the fact that the forms make up a tree structure in which successive embedded sections make branches while fields – leaves of the tree. Thus it is possible to process a form with the use of tree-searching algorithms. To fulfil this task the depth-first search algorithm was selected. This algorithm is the most compliant with the way people fill real forms. The functioning principle of the dialogue module is presented below in a simplified way. While analyzing successive steps one should remember about the iteration-recurrent nature of the algorithm.

1. At the beginning of each dialogue , the dialogue module detects a proper form class in the form ontology and creates its instance.
2. The newly created instance becomes a currently processed node and, at the same time, the top of the tree.
3. For the currently processed node, all required and optional properties are found and sorted according to the arrangement determined in the ontology. Then the successively found properties are processed.
4. If the currently processed property points at a section of the form, the section is then processed recursively.
5. In the course of processing the successive properties of the node, the dialogue module tries to detect , in the set of input data, the best adjustments both for the single fields of the form and for the whole sections. The dialogue module asks the user questions only in situations when the desired values do not exist or when it is possible to adjust more than one value to one field or section.

In order to illustrate the functioning of the dialogue module let us check how the module works on a simple form. Assuming that it is necessary to fill the form as follows:

Form:

Personal data of the user:

Name;

Surname;

Address:

City;

Street;

House No;

Aim of submitting the form;

If the process of filling the form is invoked for the first time, the dialogue with the user will look as follows:

**System:** Enter "Name".

**User:** John

**S:** Enter "Surname".

**U:** Brown

**S:** Enter "City".

**U:** London

**S:** Enter "Street".

**U:** Portland Place

**S:** Enter "House No".

**U:** 47

**S:** Enter "Aim of submitting the form".

**U:** Application for the permission to use a company car

After filling and accepting the form, the data from the form are stored in the user's profile. At the next interaction between the user and the module, the dialogue will look as follows:

**S:** I have found personal data for "John Brown" Shall I apply them (Y/N)?

**U:** Y

**S:** Enter "Aim of submitting the form".

**U:** Application to access financial data.

As one can see in the above successive filling of the form, the number of necessary interactions with the user is 6 times smaller. The reduction was possible due to the fact that the dialogue module was able to automatically fill the fields with personal data. Asking the question in the field "Aim of submitting the form" results from the updating policy of the user's profile – this issue, however, is beyond the scope of the paper. Please note that the functioning of the module is based on the real meaning of data defined in an ontology which allows to move data between different forms using common concepts defined in the shared data ontologies.

## 5 Conclusions

The use of semantic technologies allowed to implement the working software which supports the user in filling any kinds of e-forms. As both R&D and implementation works are underway the usability of the applied solution is assessed more subjectively, based on ergonomics and interaction with some forms, and is not yet supported by statistical data. However, the results obtained up until now show that the direction of the conducted works should be successful.

It is worth mentioning that the solution to the discussed problem cannot be achieved with the use of classic IT solutions which are the basis of most of today's IT systems (relational data bases, XML) and shows the potential which lies in semantic techniques.

## References

1. Berners-Lee T., Hendler J., Lassila O., *The Semantic Web*, Scientific American [http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21\\_](http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21_)
2. W3C, *Resource Description Framework*, <http://www.w3.org/RDF>
3. W3C, *OWL Web Ontology Language*, <http://www.w3.org/TR/owl-features/>
4. Horrocks I., Patel-Schneider P. F., Boley H., Tabet S., Grosf B., Dean M., *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, <http://www.w3.org/Submission/SWRL/>
5. Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y., *Pellet: A Practical OWL-DL Reasoner*, <http://pellet.owldl.com/papers/sirin07pellet.pdf>
6. Parsia B., Sirin E., Grau B. C., Ruckhaus E., Hewlett D., *Cautiously Approaching SWRL*, <http://www.mindswap.org/papers/CautiousSWRL.pdf>
7. HP Labs, *Jena – A Semantic Web Framework for Java*, <http://jena.sourceforge.net/>
8. Horrocks I., Parsia B., Patel-Schneider P., Hendler J., *Semantic web architecture: Stack or two towers?*, <http://www.cs.man.ac.uk/~horrocks/Publications/download/2005/HPPH05.pdf>
9. RoboForm - <http://www.roboform.com>
10. Mojo Navigator - <http://www.mozilla.org/projects/ui/communicator/browser/formfill/>
11. <http://gospodarka.gazeta.pl/podatki/1,25044,3963883.html>